

UJIAN AKHIR SEMESTER PBO
PONG GAME

Mata Kuliah
Pemrograman Berorientasi Object

Oleh
Ulur Rosyad Rachmandani
24091397100
2024C



PROGRAM STUDI D4 MANAJEMEN INFORMATIKA
FAKULTAS VOKASI
UNIVERSITAS NEGERI SURABAYA
2025

DAFTAR ISI

DAFTAR ISI	2
BAB I PENDAHULUAN	4
1.1. Latar Belakang	4
1.2. Tujuan Project	4
1.3. Ruang Lingkup.....	4
BAB II ANALISIS DAN PERANCANGAN SISTEM.....	5
2.1. Analisis Kebutuhan.....	5
2.1.1 Kebutuhan Functional.....	5
2.1.2 Kebutuhan Non-Functional.....	5
2.2. Perancangan Class	6
2.2.1 Class Diagram.....	6
2.2.2 Hubungan Antar Class	7
2.3. Perancangan User Interface	7
2.4.1 Screen States	7
2.4.2 Color Scheme.....	9
BAB III FITUR DAN CARA BERMAIN GAME	10
3.1. Deskripsi Game.....	10
3.2. Fitur Utama Gam	10
3.2.1 Fitur Inti Gameplay.....	10
3.2.1 Fitur Lanjutan.....	12
3.3. Kontrol Permaina	16
3.3.1 Kontrol Pemain	16
3.3.2 Kontrol Game.....	16
3.4. Cara Bermain	16
3.4.1 Memulai Game.....	16
3.4.2 Gameplay Loop.....	17
BAB IV IMPLEMENTASI SISTEM	18
4.1. Implementasi Encapsulation	18
4.2. Implementasi Inheritance	19
4.3. Implementasi Polymorphism	20
4.4. Implementasi Game Loop.....	20
4.5. Implementasi Collision Detection.....	22

4.6.	Implementasi Power-up System	22
4.7.	Implementasi Particle System.....	23
4.8.	Implementasi Sound Manager	24
BAB IV	PENUTUP	25
4.1.	Kesimpulan	25
4.2.	Kendala dan Solusi	25
4.3.	Saran Pengembangan	25

BAB 1

PENDAHULUAN

1.1. Latar Belakang

Pemrograman Berorientasi Objek (Object-Oriented Programming/OOP) adalah paradigma pemrograman yang mengorganisir kode berdasarkan "objek" yang merepresentasikan entitas dalam dunia nyata. OOP memiliki empat pilar utama: *Encapsulation* (pembungkusan), *Inheritance* (pewarisan), *Polymorphism* (banyak bentuk), dan *Abstraction* (abstraksi).

Dalam project UAS ini, saya mengimplementasikan konsep-konsep OOP tersebut dalam bentuk game Pong interaktif. Game Pong dipilih karena memiliki struktur yang jelas untuk mendemonstrasikan prinsip OOP, di mana setiap komponen game (bola, paddle, power-up) dapat dimodelkan sebagai objek dengan properties dan behaviors yang berbeda.

1.2. Tujuan Project

Tujuan Pembelajaran

- Memahami dan menerapkan konsep Encapsulation dalam class design
- Mengimplementasikan Inheritance untuk code reusability
- Mendemonstrasikan Polymorphism melalui method overriding
- Membangun aplikasi kompleks dengan struktur OOP yang baik

Tujuan Aplikasi

- Membuat game Pong yang playable dan entertaining
- Mengimplementasikan collision detection yang akurat
- Menambahkan fitur-fitur menarik (power-ups, effects, sounds)
- Membuat UI yang modern dan user-friendly

1.3. Ruang Lingkup

Project ini mencakup:

- Implementasi 5 class utama dengan hierarki inheritance
- Sistem game loop dengan 60 FPS
- Collision detection untuk bola, paddle, dan power-ups

- Particle system untuk visual effects
- Sound system untuk audio feedback
- Menu system dengan state management

BAB II

ANALISIS DAN PERANCANGAN SISTEM

2.1. Analisis Kebutuhan

2.1.1 Kebutuhan Functional

Gameplay Requirements:

- Game harus support 2 pemain lokal
- Paddle dapat bergerak atas/bawah dengan smooth
- Bola bergerak otomatis dan memantul pada collision
- Sistem scoring yang akurat
- Win condition saat mencapai skor tertentu

Feature Requirements:

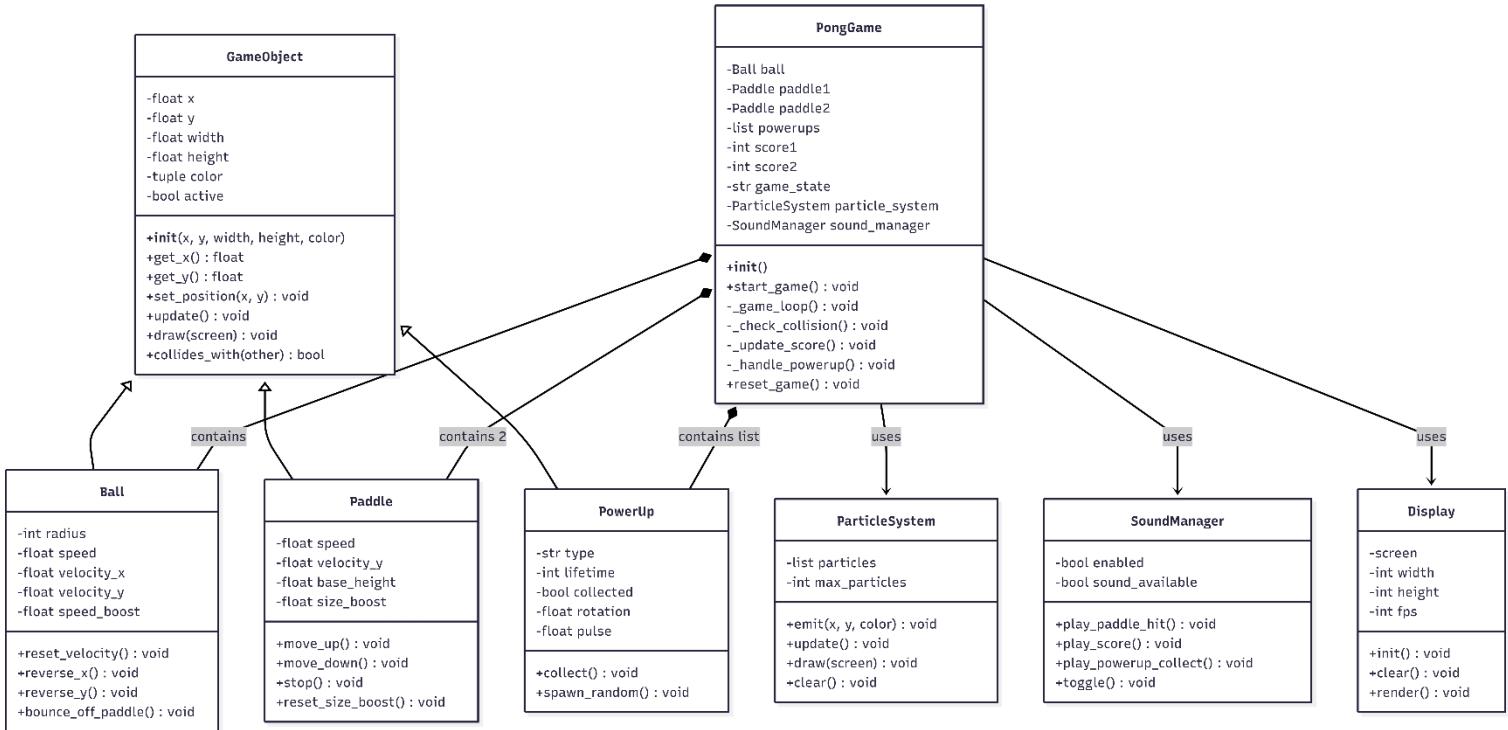
- Power-up system (speed boost, size boost)
- Particle effects untuk visual feedback
- Sound effects untuk audio feedback
- Menu system (start, pause, game over)

2.1.2 Kebutuhan Non-Functional

- **Performance:** Game harus berjalan minimal 60 FPS
- **Usability:** Kontrol responsif dan intuitif
- **Reliability:** Tidak ada bug atau crash saat gameplay
- **Maintainability:** Kode terstruktur dan terdokumentasi

2.2. Perancangan Class

2.2.1 Class Diagram



Penjelasan Class Diagram:

a. GameObject (Parent Class)

Attributes: `x`, `y`, `width`, `height`, `color`, `active`

Methods: `get/set` methods, `update()`, `draw()`, `collides_with()`

Role: Base class untuk semua objek game

b. Ball (Child of GameObject)

Additional Attributes: `radius`, `speed`, `velocity_x`, `velocity_y`, `speed_boost`

Additional Methods: `reset_velocity()`, `reverse_x()`, `reverse_y()`, `bounce_off_paddle()`

Role: Representasi bola dalam game

c. Paddle (Child of GameObject)

Additional Attributes: `speed`, `velocity_y`, `base_height`, `size_boost`

Additional Methods: `move_up()`, `move_down()`, `stop()`, `reset_size_boost()`

Role: Representasi paddle yang dikontrol pemain

d. PowerUp (Child of GameObject):

Additional Attributes: type, lifetime, collected, rotation, pulse

Additional Methods: collect(), spawn_random()

Role: Item power-up yang memberikan efek khusus

e. PongGame (Game Manager):

Attributes: ball, paddle1, paddle2, score1, score2, game_state

Methods: start_game(), _game_loop(), _check_collision(), _update_score()

Role: Mengatur seluruh logika dan flow game

f. ParticleSystem:

Attributes: particles[]

Methods: emit(), update(), draw()

Role: Mengelola efek visual partikel

g. SoundManager:

Attributes: enabled, sound_available

Methods: play_paddle_hit(), play_score(), play_powerup_collect()

Role: Mengelola sound effects

2.2.2 Hubungan Antar Class

Inheritance Relationships:

- GameObject → Ball (IS-A relationship)
- GameObject → Paddle (IS-A relationship)
- GameObject → PowerUp (IS-A relationship)

Composition Relationships:

- PongGame contains Ball (HAS-A relationship)
- PongGame contains Paddle (HAS-A relationship)
- PongGame contains PowerUp (HAS-A relationship)
- PongGame contains ParticleSystem (HAS-A relationship)
- PongGame contains SoundManager (HAS-A relationship)

2.3. Perancangan User Interface

2.4.1 Screen States

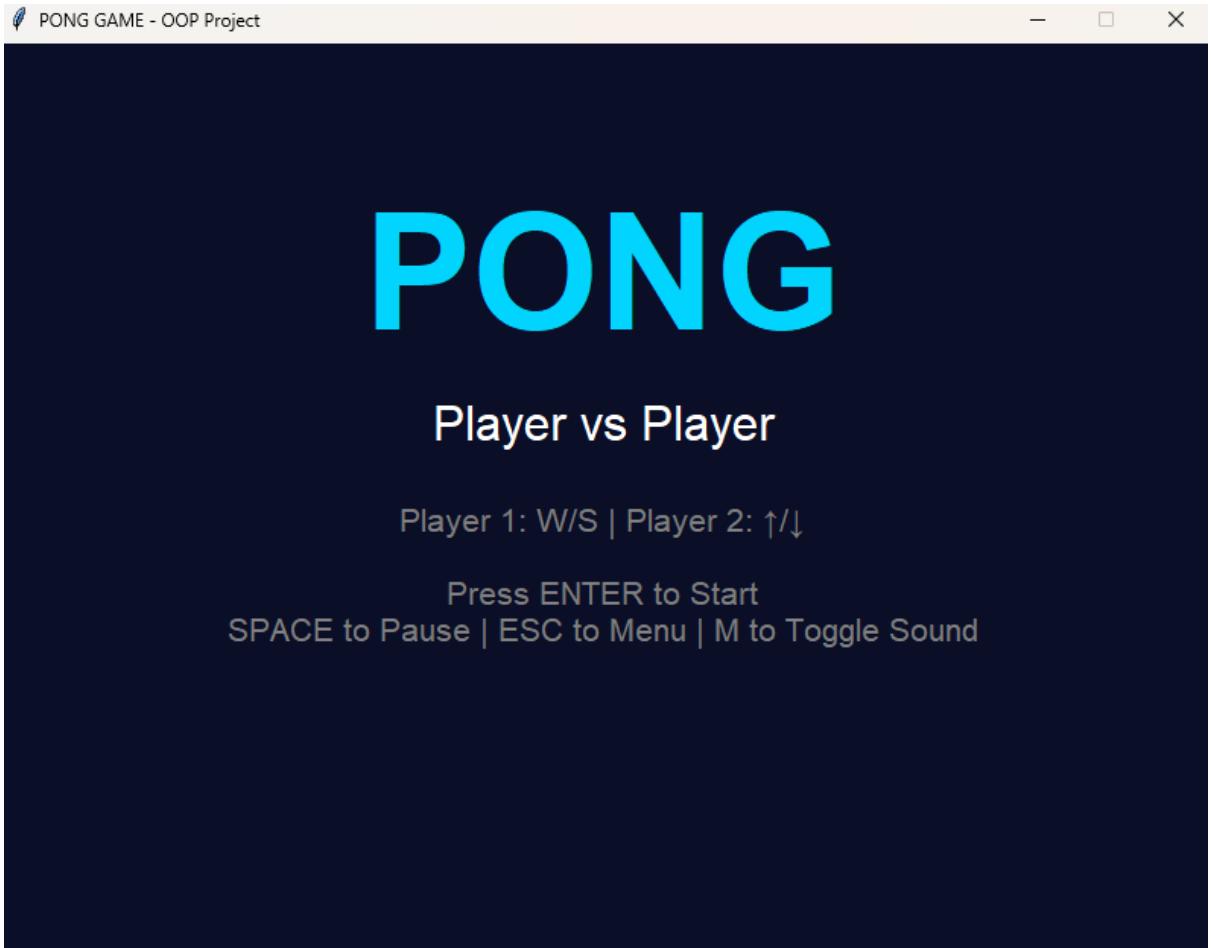
Menu State:

Title: "PONG"

Subtitle: "Player vs Player"

Instructions: Controls dan cara bermain

Prompt: "Press ENTER to Start"



Playing State:

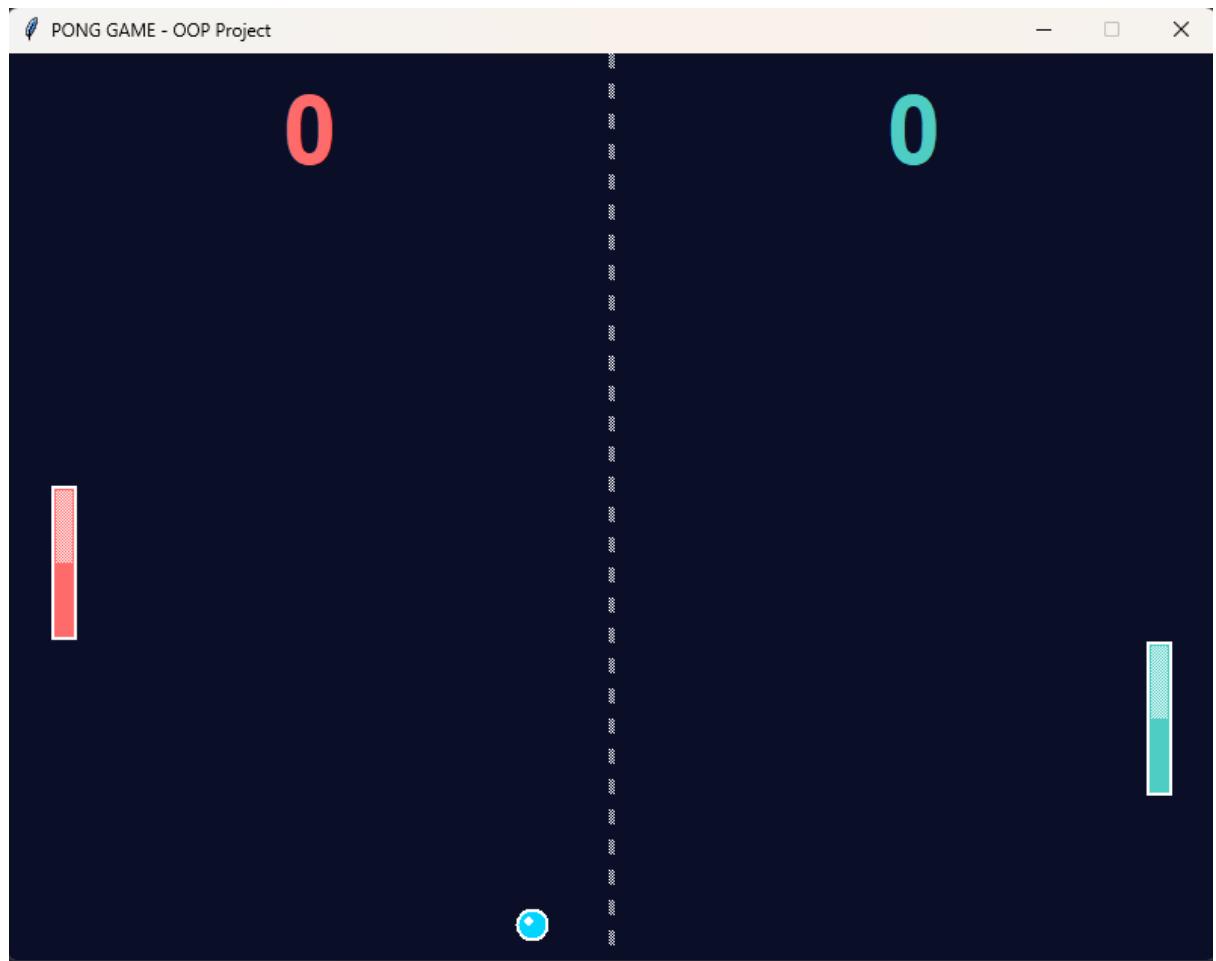
Canvas dengan background hitam

Paddle kiri (merah) dan kanan (cyan)

Bola di tengah

Score display di atas

Center line putih



Game Over State:

Winner announcement

Final scores

Options: Play again atau back to menu

2.4.2 Color Scheme

Background: #0a0e27 (Dark blue)

Accent: #00d4ff (Cyan)

Player 1: #FF6B6B (Red)

Player 2: #4ECDC4 (Teal)

Text: White

BAB III

FITUR DAN CARA BERMAIN GAME

3.1. Deskripsi Game

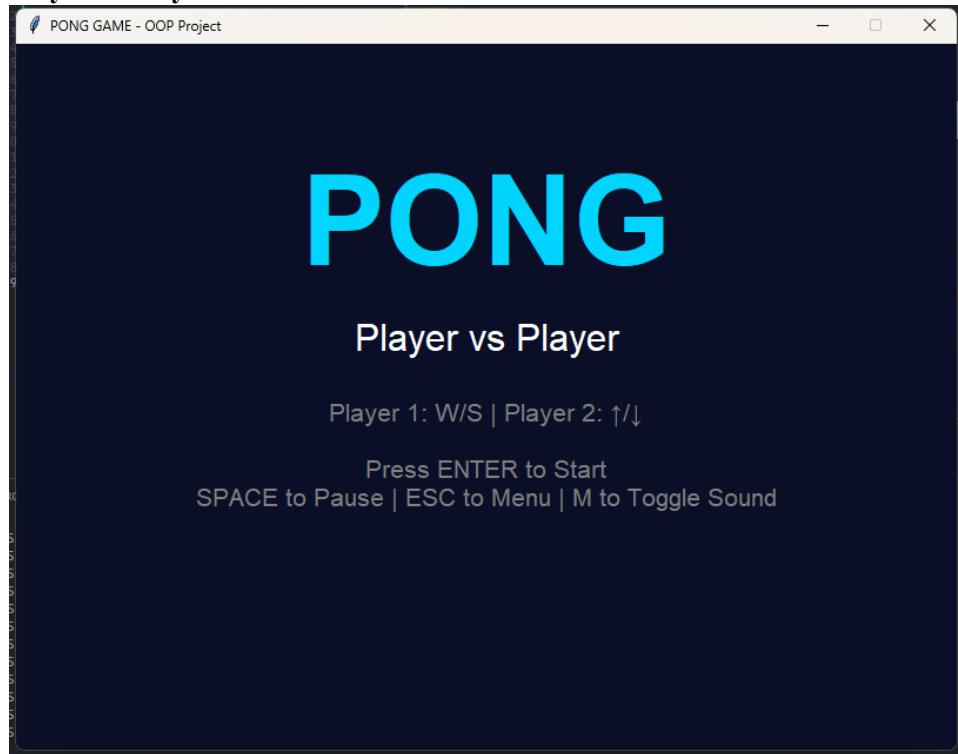
Pong adalah game klasik two-player di mana setiap pemain mengontrol sebuah paddle vertikal untuk memantulkan bola. Tujuan utama adalah mencetak poin dengan membuat bola melewati paddle lawan. Pemain pertama yang mencapai 5 poin memenangkan permainan.

Game ini mengimplementasikan mekanik game klasik Pong dengan penambahan fitur modern seperti power-ups, particle effects, dan sound effects untuk meningkatkan pengalaman bermain.

3.2. Fitur Utama Game

3.2.1 Fitur Inti Gameplay

a. Player vs Player Mode



Mode permainan lokal untuk dua pemain yang bermain di komputer yang sama.

Karakteristik:

- Pemain 1 (kiri) menggunakan keyboard W/S
- Pemain 2 (kanan) menggunakan arrow keys ↑↓
- Real-time competitive gameplay
- Fair play dengan speed dan size paddle yang sama

b. Dynamic Ball Physics

Bola bergerak dengan physics realistik dan angle-based bouncing.

Karakteristik:

- Kecepatan awal: 5 pixels per frame
- Arah random saat spawn (kiri/kanan dengan angle $\pm 45^\circ$)
- Bounce angle dipengaruhi posisi impact pada paddle
- Speed gradually increases setelah beberapa hit

Formula Bounce Angle:

$$\text{Impact Position} = (\text{Ball Y} - \text{Paddle Center Y}) / (\text{Paddle Height} / 2)$$

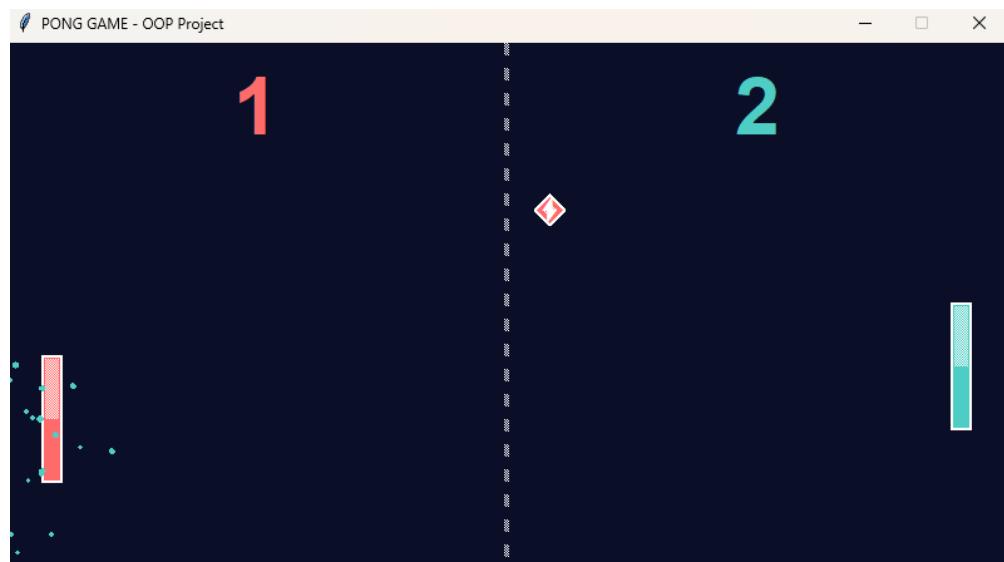
$$\text{Bounce Angle} = \text{Impact Position} \times 60^\circ \text{ (maksimal)}$$

Example:

- Hit di tengah paddle → Bounce horizontal (0°)
- Hit di ujung atas paddle → Bounce diagonal ke atas (60°)
- Hit di ujung bawah paddle → Bounce diagonal ke bawah (-60°)

c. Scoring System

Scoring system adalah sistem yang digunakan untuk perhitungan skor otomatis dengan winning condition.



Karakteristik:

- +1 poin jika bola melewati paddle lawan
- Winning score: 5 poin
- Score display real-time di bagian atas layar
- Color-coded scores (merah untuk P1, cyan untuk P2)

Flow:

Ball keluar kiri → Player 2 score +1

Ball keluar kanan → Player 1 score +1

Score = 5 → Game Over

d. Collision Detection

Sistem deteksi tabrakan akurat menggunakan AABB (Axis-Aligned Bounding Box).

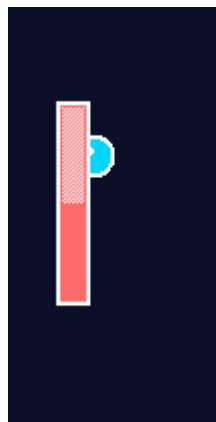
Jenis Collision:

1. Ball vs Wall (atas/bawah)



- Membalikkan bola dengan velocity Y
- Memberikan sound effect
- Mengeluarkan particles percikan

2. Ball vs Paddle



- Mengkalkulasi angle pantulan
- Membalikkan bola dengan velocity Y
- Memberikan sound effect
- Mengeluarkan particles percikan

3. Ball vs PowerUp

- Mendapatkan dan menambah Power Up
- Menghasilkan efek partikel

3.2.1 Fitur Lanjutan

a. Power-Up System

Item special yang muncul secara random dan memberikan efek sementara.

Tipe Power-Ups:

1. Speed Boost ⚡



- Icon: Lightning bolt (⚡)
- Warna: Merah (#FF6B6B)
- Efek: Meningkatkan kecepatan bola 50% (1.5x)
- Durasi: 3 detik (180 frames @ 60 FPS)
- Strategy: Bagus untuk pressure opponent dengan serangan cepat

2. Size Boost ↑



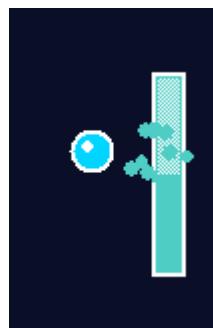
- Icon: Up arrow (↑)
- Warna: Cyan (#4ECDC4)
- Efek: Memperbesar paddle 50% (1.5x)
- Durasi: 3 detik (180 frames @ 60 FPS)
- Strategy: Bagus untuk defense, lebih mudah block bola

b. Particle Effects System

Visual feedback dinamis saat terjadi event penting dalam game.

Jenis Particle Effects:

1. Paddle Hit Particles



- Trigger: Ball collision dengan paddle
- Jumlah: 10 particles
- Warna: Sesuai warna paddle (merah/cyan)
- Behavior: Explode outward dengan random direction
- Lifetime: 20-40 frames

2. Wall Hit Particles



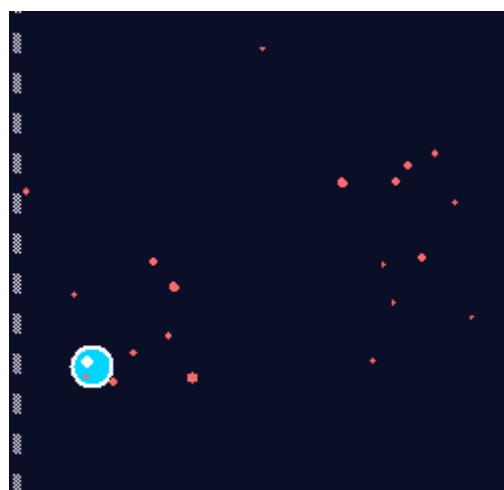
- Trigger: Ball collision dengan wall atas/bawah
- Jumlah: 8 particles
- Warna: Accent color (cyan)
- Behavior: Horizontal spread
- Lifetime: 15-30 frames

3. Score Particles



- Trigger: Ball keluar dari arena (scoring)
- Jumlah: 30 particles
- Warna: Sesuai pemain yang score
- Behavior: Large explosion effect
- Lifetime: 30-50 frames

4. Power-Up Collection Particles



- Trigger: Ball mengambil power-up
- Jumlah: 20 particles

- Warna: Sesuai warna power-up
- Behavior: Burst effect
- Lifetime: 25-40 frames

c. Sound Effects System

Adalah fitur audio feedback untuk setiap action dalam game

Jenis Sound Effects:

1. Paddle Hit Sound

- Frequency: 800 Hz
- Duration: 50 ms
- Character: Sharp, high-pitched beep
- Trigger: Ball hits paddle

2. Wall Hit Sound

- Frequency: 600 Hz
- Duration: 50 ms
- Character: Medium-pitched beep
- Trigger: Ball hits wall

3. Score Sound

- Sequence: 400 Hz (100ms) → 300 Hz (150ms)
- Character: Descending double beep
- Trigger: Player scores

4. Power-Up Collection Sound

- Sequence: 500 Hz → 700 Hz → 900 Hz (50ms each)
- Character: Ascending triple beep (positive)
- Trigger: Ball collects power-up

5. Game Start Sound

- Sequence: 600 Hz (100ms) → 800 Hz (150ms)
- Character: Ascending double beep
- Trigger: Game starts

6. Game Over Sound

- Sequence: 800 Hz → 600 Hz → 400 Hz (100-200ms)
- Character: Descending triple beep (ending)

- Trigger: Player wins
- Sound Toggle: Tekan M untuk enable/disable sound effects.

3.3. Kontrol Permaina

3.3.1 Kontrol Pemain

Pemain	Kontrol	Fungsi	Note
Player 1	W atau w	Gerakkan paddle ke atas	Support uppercase & lowercase
Player 1	S atau s	Gerakkan paddle ke bawah	Support uppercase & lowercase
Player 2	↑ (Arrow Up)	Gerakkan paddle ke atas	-
Player 2	↓ (Arrow Down)	Gerakkan paddle ke bawah	-

3.3.2 Kontrol Game

Tombol	Fungsi	Context	Behavior
ENTER	Mulai game	Menu screen	Start new game
ENTER	Main lagi	Game over screen	Restart dengan score 0-0
SPACE	Pause	Saat bermain	Freeze game state
SPACE	Resume	Saat pause	Continue game
ESC	Ke menu	Any state	Return to main menu
M	Toggle sound	Any state	Enable/disable sound effects

3.4. Cara Bermain

3.4.1 Memulai Game

Langkah-langkah:

1. Jalankan Aplikasi
2. Menu Screen Muncul
 - Tampilan title "PONG"
 - Instruksi controls
 - Mode permainan
3. Tekan ENTER
 - Game langsung dimulai
 - Ball spawn di tengah dengan arah random
 - Score dimulai dari 0-0

3.4.2 Gameplay Loop

Basic Gameplay:

1. Ball Movement

- Ball bergerak otomatis dengan kecepatan konstan
- Arah ditentukan saat spawn atau bounce

2. Paddle Control

- Gerakkan paddle untuk memantulkan ball
- Posisi impact menentukan bounce angle
- Jangan biarkan ball melewati paddle Anda

3. Scoring

- Jika ball melewati paddle kiri → Player 2 +1 poin
- Jika ball melewati paddle kanan → Player 1 +1 poin
- Ball reset ke tengah setelah scoring

4. Power-Ups (Optional)

- Power-up muncul random di tengah arena
- Arahkan ball untuk menyentuh power-up
- Dapatkan efek special selama 3 detik

5. Winning

- First to 5 points wins
- Game over screen muncul
- Option: Play again atau back to menu

BAB IV

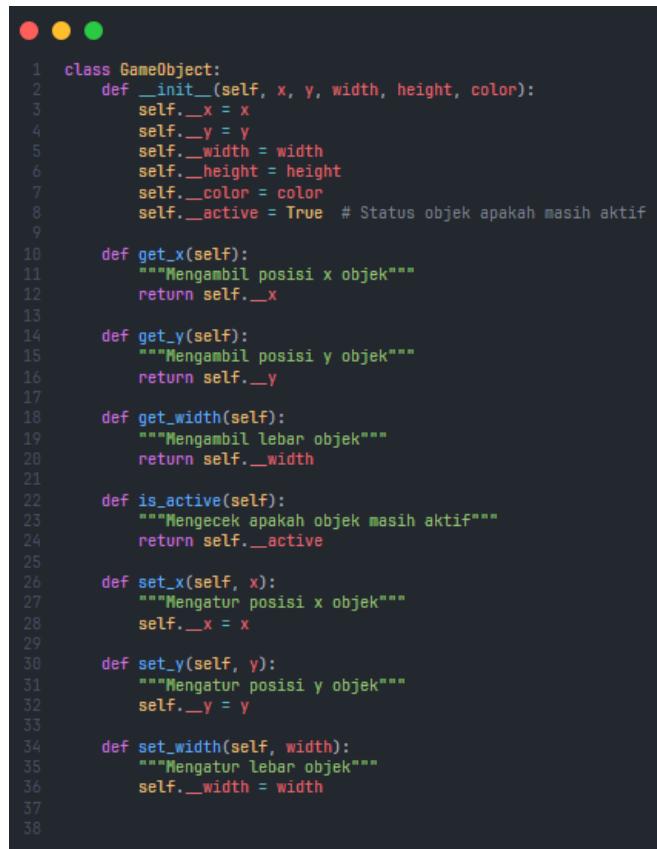
IMPLEMENTASI SISTEM

4.1. Implementasi Encapsulation

Encapsulation adalah proses menyembunyikan detail implementasi internal suatu objek dan hanya menyediakan interface untuk berinteraksi dengan objek tersebut. Dalam Python, encapsulation dilakukan dengan:

- Private attributes: menggunakan prefix `__` (double underscore)
- Getter methods: untuk mengakses nilai private attributes
- Setter methods: untuk memodifikasi nilai private attributes

Contoh di class GameObject:



```
● ● ●
1  class GameObject:
2      def __init__(self, x, y, width, height, color):
3          self.__x = x
4          self.__y = y
5          self.__width = width
6          self.__height = height
7          self.__color = color
8          self.__active = True # Status objek apakah masih aktif
9
10     def get_x(self):
11         """Mengambil posisi x objek"""
12         return self.__x
13
14     def get_y(self):
15         """Mengambil posisi y objek"""
16         return self.__y
17
18     def get_width(self):
19         """Mengambil lebar objek"""
20         return self.__width
21
22     def is_active(self):
23         """Mengecek apakah objek masih aktif"""
24         return self.__active
25
26     def set_x(self, x):
27         """Mengatur posisi x objek"""
28         self.__x = x
29
30     def set_y(self, y):
31         """Mengatur posisi y objek"""
32         self.__y = y
33
34     def set_width(self, width):
35         """Mengatur lebar objek"""
36         self.__width = width
37
38
```

Keuntungan:

- Data terlindungi dari akses langsung
- Dapat menambahkan validation di setter
- Flexibility untuk mengubah implementasi internal tanpa mengubah interface

4.2. Implementasi Inheritance

Inheritance adalah mekanisme di mana sebuah class (child class) dapat mewarisi attributes dan methods dari class lain (parent class). Ini memungkinkan code reusability dan hierarki class yang logis.

Contoh di class turunan dari GameObject:



```
 1  class GameObject:
 2      def __init__(self, x, y, width, height, color):
 3          self._x = x
 4          self._y = y
 5          self._width = width
 6          self._height = height
 7          self._color = color
 8          self._active = True
 9
10  class Ball(GameObject):
11      def __init__(self, x, y, radius, color, speed):
12          super().__init__(x, y, radius * 2, radius * 2, color)
13          self._radius = radius
14          self._speed = speed
15          self._base_speed = speed
16          self._velocity_x = 0
17          self._velocity_y = 0
18          self._speed_boost = 1.0
19          self._reset_velocity()
20
21  class Paddle(GameObject):
22      def __init__(self, x, y, width, height, color, speed):
23          super().__init__(x, y, width, height, color)
24
25          self._speed = speed
26          self._base_height = height
27          self._velocity_y = 0
28          self._size_boost = 1.0
29          self._screen_height = 600
30
31  class PowerUp(GameObject):
32      SPEED_BOOST = "speed_boost"
33      SIZE_BOOST = "size_boost"
34
35      COLORS = {
36          SPEED_BOOST: "#FF6B6B",
37          SIZE_BOOST: "#4ECDC4"
38      }
39      def __init__(self, x, y, size, powerup_type):
40          color = self.COLORS.get(powerup_type, "#FFFFFF")
41          super().__init__(x, y, size, size, color)
42          self._type = powerup_type
43          self._size = size
44          self._rotation = 0
45          self._pulse = 0
46          self._lifetime = 300
47          self._collected = False
48
```

Keuntungan:

- Code reusability: tidak perlu menulis ulang attributes dan methods umum
- Hierarki yang jelas: semua game objects berbagi interface yang sama

- Extensibility: mudah menambahkan objek game baru

4.3. Implementasi Polymorphism

Polymorphism adalah kemampuan objek yang berbeda untuk merespon method yang sama dengan cara yang berbeda. Dalam Python, ini sering dilakukan dengan method overriding.

Method update() di berbagai class:



```

1  class GameObject:
2      def update(self):
3          pass;
4
5  class Ball(GameObject):
6      def update(self):
7          # Update posisi berdasarkan velocity
8          self.set_x(self.get_x() + self.__velocity_x)
9          self.set_y(self.get_y() + self.__velocity_y)
10
11 class Paddle(GameObject):
12     def update(self):
13         if not self.is_active():
14             return
15
16         # Update posisi y berdasarkan velocity
17         new_y = self.get_y() + self.__velocity_y
18
19         # Boundary checking - pastikan paddle tidak keluar layar
20         if new_y < 0:
21             new_y = 0
22         elif new_y + self.get_height() > self.__screen_height:
23             new_y = self.__screen_height - self.get_height()
24
25         self.set_y(new_y)
26

```

Keuntungan:

- Setiap objek dapat di-update dengan cara yang sesuai karakteristiknya
- Game loop menjadi lebih simple dan generic
- Mudah menambahkan objek baru tanpa mengubah game loop

4.4. Implementasi Game Loop

Game loop adalah jantung dari game yang berjalan setiap frame (60 FPS) yang merupakan pola fundamental dalam game development, game loop disini terdiri dari:

- Input Processing: Membaca input dari user
- Update: Update state game (posisi, collision, dll)

- Render: Menggambar frame baru ke layer

```
● ○ ●
1 def __game_loop(self):
2     if not self.__game_running:
3         return
4
5     if self.__game_state != "PLAYING":
6         self.__update_id = self.root.after(1000 // self.FPS, self.__game_loop)
7
8
```

1. UPDATE PHASE

```
● ○ ●
1 self.ball.update()
2 self.paddle1.update()
3 self.paddle2.update()
4
5 # Update power-up
6 if self.__current_powerup and self.__current_powerup.is_active():
7     self.__current_powerup.update()
8     # Jika lifetime habis, set ke None agar bisa spawn lagi
9     if not self.__current_powerup.is_active():
10         self.__current_powerup = None
11
12 self.__update_powerup()
13 self.__spawn_powerup()
14 self.__check_powerup_collection()
15
```

2. COLLISION DETECTION

```
● ○ ●
1 # Check collisions
2 self.__check_wall_collision()
3 self.__check_paddle_collision()
4 self.__check_scoring()
5
```

3. RENDER PHASE

```
● ○ ●
1 # Render everything
2 self.__render()
3
```

4. SCHEDULE NEXT FRAME



```
1 self.__update_id = self.root.after(1000 // self.FPS, self._game_loop)
```

4.5. Implementasi Collision Detection

Collision detection menggunakan AABB (Axis-Aligned Bounding Box) method. Collision detection adalah proses mendeteksi apakah dua objek saling bersentuhan. Dalam project ini menggunakan *AABB (Axis-Aligned Bounding Box)* method:



```
1 def _check_wall_collision(self):
2     """Cek collision dengan dinding atas/bawah"""
3     ball_y = self.ball.get_y()
4     ball_radius = self.ball.get_radius()
5
6     # Collision dengan dinding atas
7     if ball_y - ball_radius < 0:
8         self.ball.set_y(ball_radius)
9         self.ball.reverse_y()
10    self.sound_manager.play_wall_hit()
11    self.particle_system.emit(self.ball.get_x(), 0, self.ACCENT_COLOR, count=8)
12    return True
13
14    # Collision dengan dinding bawah
15    if ball_y + ball_radius > self.HEIGHT:
16        self.ball.set_y(self.HEIGHT - ball_radius)
17        self.ball.reverse_y()
18        self.sound_manager.play_wall_hit()
19        self.particle_system.emit(self.ball.get_x(), self.HEIGHT, self.ACCENT_COLOR, count=8)
20    return True
21
22    return False
```

4.6. Implementasi Power-up System

Power-up system terdiri dari spawning, collection, dan effect application.

```

1  def _check_powerup_collection(self):
2      """Cek apakah ada yang mengambil power-up"""
3      if self._current_powerup is None or not self._current_powerup.is_active():
4          return
5
6      # Cek collision dengan ball
7      if self.ball.collides_with(self._current_powerup):
8          powerup_type = self._current_powerup.get_type()
9
10     # Simpan posisi untuk particle effect
11     powerup_x = self._current_powerup.get_x()
12     powerup_y = self._current_powerup.get_y()
13     powerup_color = self._current_powerup.get_color()
14
15     # Collect power-up
16     self._current_powerup.collect()
17     self.sound_manager.play_powerup_collect()
18
19     # Tentukan siapa yang dapat power-up berdasarkan arah bola
20     if self.ball.get_velocity_x() > 0:
21         # Bola ke kanan, player 2 dapat power-up
22         self._powerup_active_player = 2
23         if powerup_type == PowerUp.SPEED_BOOST:
24             self.ball.set_speed_boost(1.5)
25         elif powerup_type == PowerUp.SIZE_BOOST:
26             self.paddle2.set_size_boost(1.5)
27     else:
28         # Bola ke kiri, player 1 dapat power-up
29         self._powerup_active_player = 1
30         if powerup_type == PowerUp.SPEED_BOOST:
31             self.ball.set_speed_boost(1.5)
32         elif powerup_type == PowerUp.SIZE_BOOST:
33             self.paddle1.set_size_boost(1.5)
34
35     self._powerup_timer = self._powerup_duration

```

4.7. Implementasi Particle System

Particle system untuk visual effects saat collision.

```

1  class ParticleSystem:
2      def __init__(self):
3          self._particles = []
4
5      def emit(self, x, y, color, count=10):
6          for _ in range(count):
7              particle = Particle(x, y, color)
8              self._particles.append(particle)
9
10     def update(self):
11         # Update dan hapus particle yang tidak aktif
12         self._particles = [p for p in self._particles if p.is_active()]
13
14     def draw(self, canvas):
15         for particle in self._particles:
16             particle.draw(canvas)
17
18
19

```

4.8. Implementasi Sound Manager

Sound manager menggunakan winsound (Windows) untuk beep sounds.

```
● ● ●

1 import sys
2 import threading
3 class SoundManager:
4     def __init__(self):
5         """Constructor untuk SoundManager"""
6         self.__enabled = True
7         self.__sound_available = False
8
9     try:
10         if sys.platform == 'win32':
11             import winsound
12             self.__winsound = winsound
13             self.__sound_available = True
14         else:
15             self.__sound_available = False
16     except ImportError:
17         self.__sound_available = False
18
19     def enable(self):
20         """Enable sound effects"""
21         self.__enabled = True
22
23     def disable(self):
24         """Disable sound effects"""
25         self.__enabled = False
26
27     def is_enabled(self):
28         """Mengecek apakah sound enabled"""
29         return self.__enabled
30
31     def toggle(self):
32         """Toggle sound on/off"""
33         self.__enabled = not self.__enabled
34         return self.__enabled
```

BAB IV

PENUTUP

4.1. Kesimpulan

Dari pengerjaan project game Pong ini, dapat disimpulkan bahwa semua implementasi OOP sukses dijalankan. Ketiga pilar OOP (Encapsulation, Inheritance, Polymorphism) berhasil diterapkan dengan baik Struktur kode menjadi lebih modular dan maintainable. Karena penerapan ini code reusability meningkat dengan inheritance hierarchy. Adapun kesimpulan lainnya antara lain:

Fungsionalitas Game yang Lengkap:

- Semua fitur utama berjalan dengan baik (movement, collision, scoring)
- Fitur tambahan (power-ups, effects, sounds) berhasil diimplementasikan
- Game playable dan entertaining

Pembelajaran yang Didapat:

- Pemahaman mendalam tentang OOP principles
- Skill dalam game development dengan Python
- Experience dalam software architecture design

4.2. Kendala dan Solusi

a. Kendala 1: Collision Detection Tidak Akurat

Problem: Ball kadang menembus paddle Solusi: Implementasi AABB collision detection dengan checking velocity direction

b. Kendala 2: Particle Performance

Problem: Terlalu banyak particles menyebabkan lag Solusi: Limit jumlah particles dan automatic cleanup untuk inactive particles

c. Kendala 3: Sound Blocking Game Loop

Problem: Play sound membuat game freeze sesaat Solusi: Gunakan threading untuk play sound secara asynchronous

4.3. Saran Pengembangan

Untuk pengembangan lebih lanjut, project ini bisa ditambahkan:

- Single Player Mode:
- Implementasi AI opponent dengan difficulty levels
- AI menggunakan algoritma prediction untuk mengikuti bola

Advanced Features:

- More power-up types (slow motion, multi-ball, etc)
- Ball trails effect untuk visual yang lebih menarik
- Background music dengan pygame.mixer
- Customizable paddle skins

Multiplayer Online:

- Network multiplayer menggunakan socket programming
- Matchmaking system
- Leaderboard global
- Better UI/UX:
 - Settings menu untuk customize controls dan audio
 - Smooth transitions between screens
 - More elaborate animations